Build Order Problem

Problem Statement

You are given a text file as an input. The structure of test file is briefed in section Input File Structure. This file has information about a set of software products and their respective dependencies on other software products. By looking at this input file, we need to figure out a build order for all the products listed in this file.

Write a C++ program which takes above mentioned input file as an input and outputs following two pieces of information

1. Was there any cyclic dependency?
2. A build order for the products. Meaning of build order is defined in the section Build Order

Example Input File

```
4
Office Word Excel
Word boost graph
Excel ace graph
graph
```

Input File Structure

Input file is a text file. Each line ends with a new line character '\n'.

- First line is an unsigned integer representing number of entries (excluding the first line) present in the file. In the Example Input File, the first line indicates that the total number of entries is 4.
- Second line and onwards has one or more space separated words. First word represents the name of a software product and second words onwards (in the same line) represent the name of dependency. In the Example Input File, second line specifies that a product named 'Office' depends on two products namely, 'Word' and 'Excel'.
- A product may specify another product as its dependency, which is defined somewhere else in the file. In the Example Input File, Office lists Word and Excel as its dependency, which are defined as products in line 3 and 4 respectively.
- A product may specify another product as its dependency, which is defined no-where else in the file. . In the Example Input File, Word lists boost and graph as its dependency, but only graph is defined (in line 5) and boost is defined nowhere in the file as a product. Boost is only referenced as a dependency.

Build Order

A build order is the order in which products should be built. The rules of building a product are simple and listed below

1. If a product has a set of dependencies, then their dependencies must be built first.
   Example: If a product P has two dependencies D1 and D2, then D1 and D2 must be built before product P
2. If a product P has two dependencies D1 and D2, then D1 and D2 must be built first. And among D1 and D2, we are free chose either of D1 or D2 to be built first, unless conflicted by dependencies of D1 or D2, if any.

Constraints

You are free to design the application in whatever ways you would want it to. However, please follow below guidelines

1. You must use standard C++ for this assignment. Non-standard extension to C++ is best avoided.
2. STL can be used
3. Key functions must specify runtime and memory complexity as comments.
4. Any compiler intrinsic or compiler extensions are not allowed
5. Any third-party library or framework (e.g., boost, ACE, MFC, QT, POCO, etc.) is not allowed. Sufficient care are has been taken in order to ensure that the solution would not require any third-party library/framework.

Guidelines for sharing the solution

1. Prepare all sources in one single file: dump your entire source in one single file named solution.txt. Please note the extension. Its 'txt' and not 'cpp' or 'hpp' or 'h'. Please make sure that the declaration and definition appears in correct order and file can be compiled successfully without making a single line of change in the contents of this file.
2. First line should be a comment specifying the C++ standard this solution assumes
3. Second line must specify the build command for this file, preferably using gcc or clang.
4. Third line and onwards should be the source code for the solution.
5. solution.txt thus prepared must be sent as a plain text attachment (do not change extension or try to zip it).